C11L12 WFI

802.11/Wi-Fi®应用开发

课程安排

- 什么是嵌入式系统的Wi-Fi®？
- 如何使用Microchip的Wi-Fi解决方案
- 如何创建具有Wi-Fi功能的应用
- 现场演示
- 客户机与服务器应用

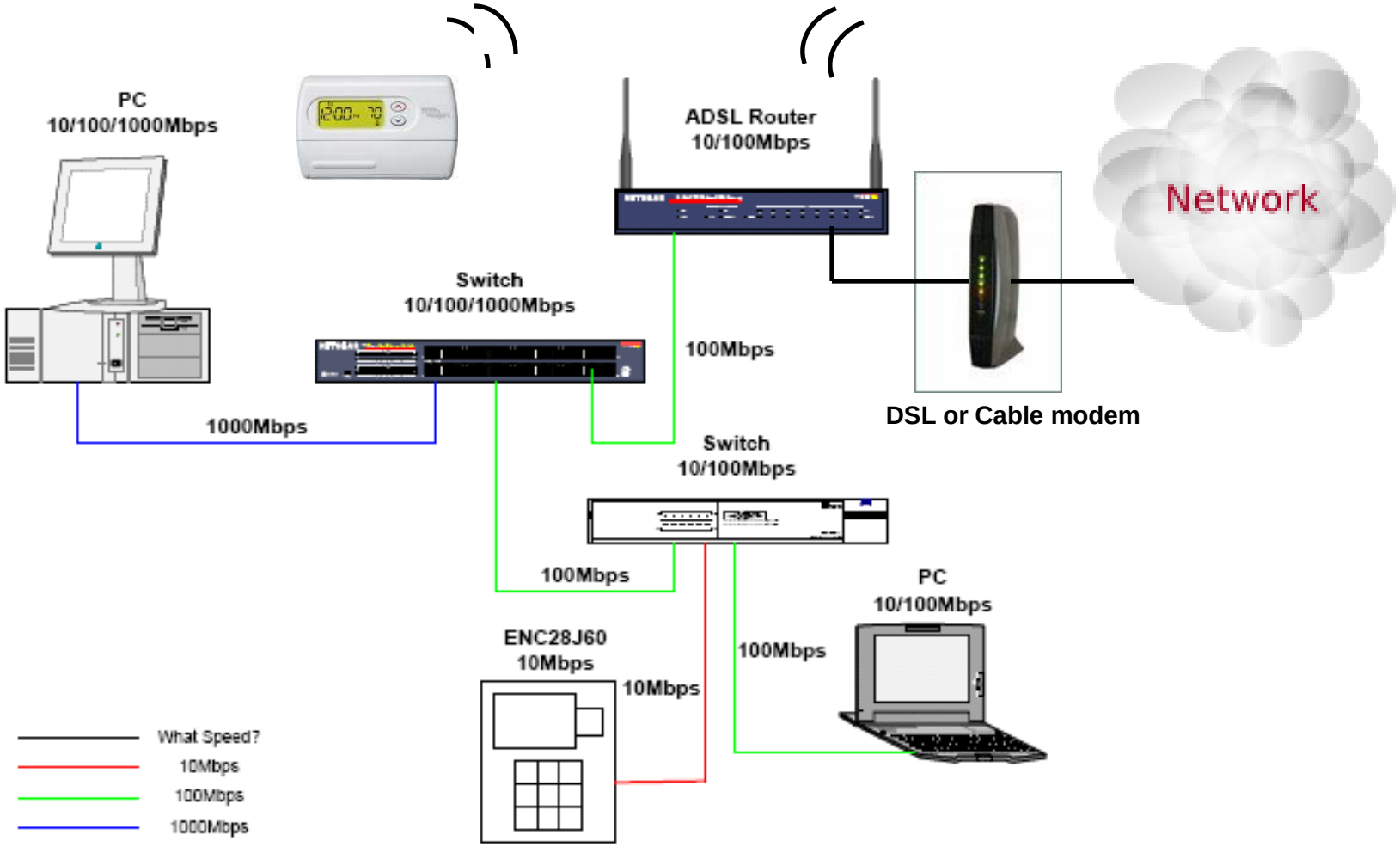
课程目标

- 当您完成本课程时，您将：
  - 了解Microchip 802.11无线解决方案
  - 能够运行和修改现成演示
  - 可以使用Microchip的开发环境，快速创建具有Wi-Fi®连接功能的应用
  - 能够在现有的嵌入式产品中设计加入Wi-Fi

嵌入式Wi-Fi®

互联网

Wi-Fi®是什么？



Wi-Fi®是什么？

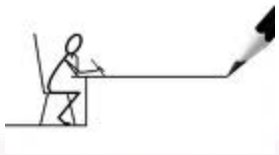
- 以太网是世界上分布最广的数据通信网络
- **Wi-Fi是无线的以太网**
  - 增加了移动联网的特性
  - 无需线缆，但保留LAN、WAN和WWW连接

嵌入式系统的Wi-Fi®

- 嵌入式产品要求
  - 对8/16位处理器的支持
  - 存储容量小（小尺寸的RAM及闪存）
    - 可以在单片机上直接运行
  - 低功耗
    - 电池供电
  - 快速推向市场

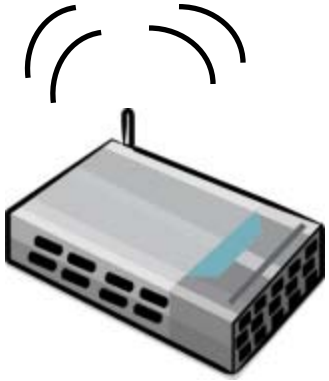
开发时希望

- 对标准的无缝支持
- 不需要互联网的专门知识
- 着手时有现成的示例可用
- 更高层次的应用模块
- 调研时投资小



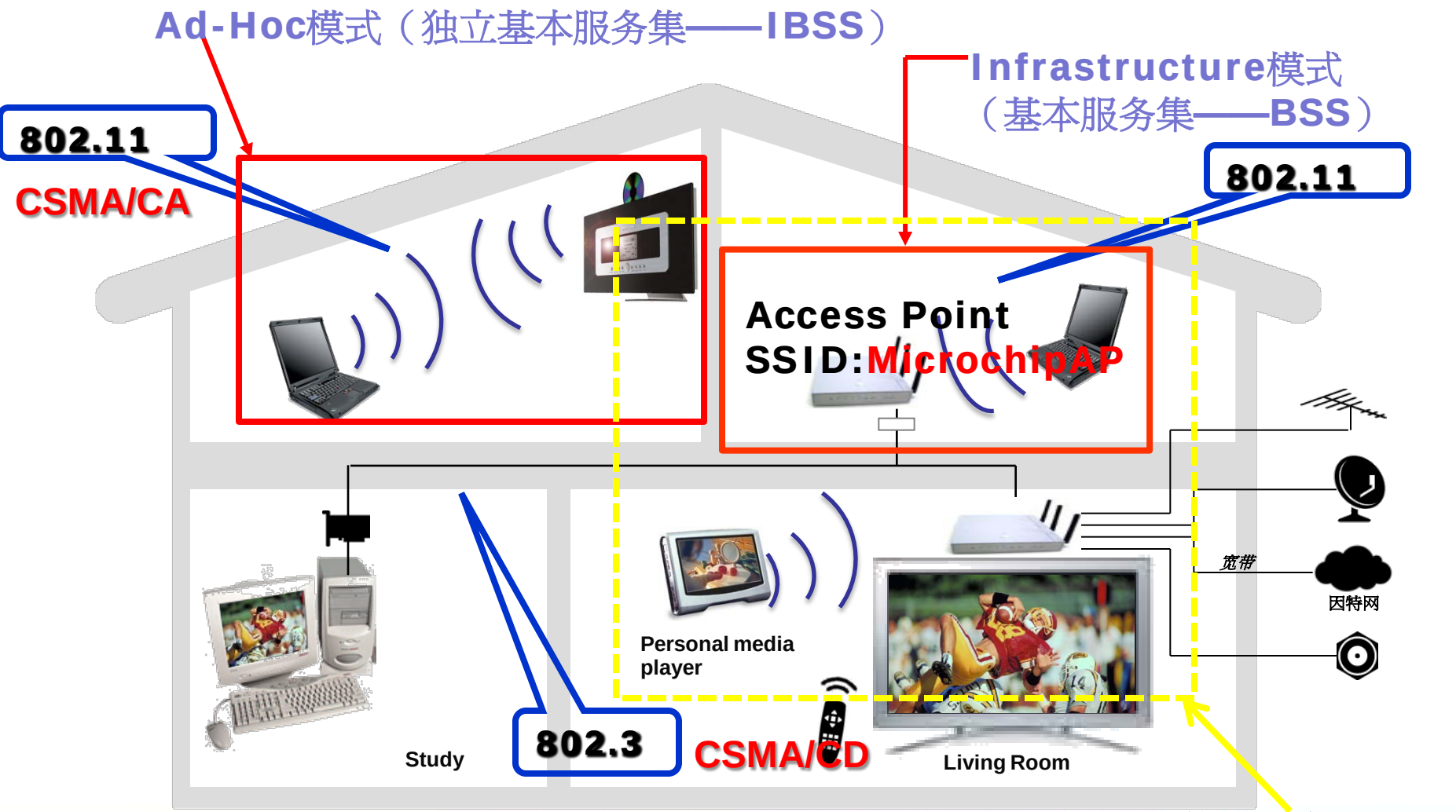
什么是802.11 Wi-Fi®

- 手提电脑
- 接入点
- 智能手机



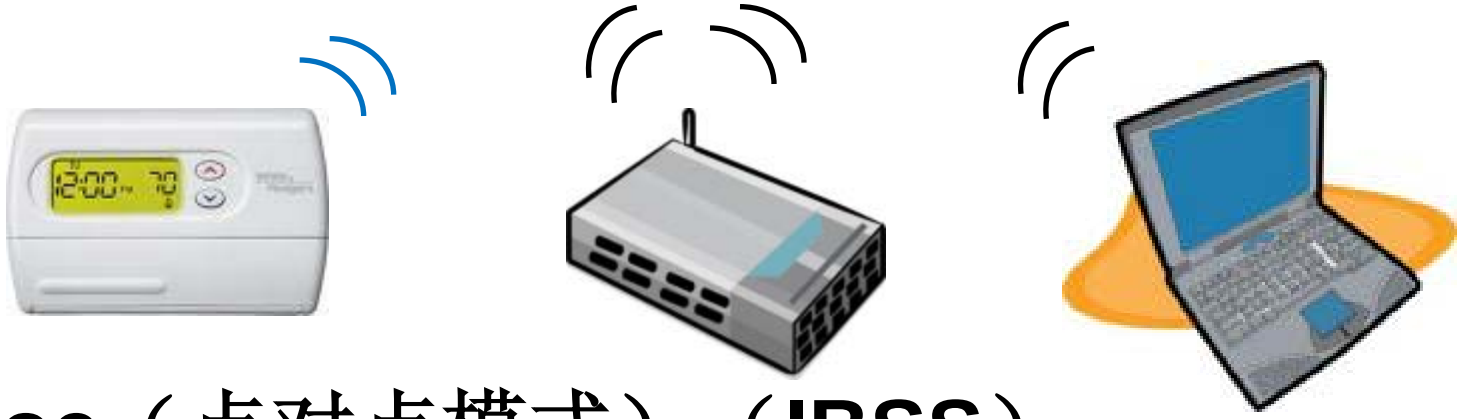
- 长距范围
- 高带宽
- 下载式MAC处理
- 用于特定硬件平台的驱动程序

什么是802.11 Wi-Fi®



Microchip 802.11解决方案

- **Infrastructure（基础设施模式）（BSS）**



- **Ad Hoc（点对点模式）（IBSS）**

Microchip 802.11解决方案

- **Wi-Fi®针对低资源的嵌入式应用**
- **高度集成、经过认证的模块解决方案**
  - 包括MAC、BaseBand、RF和PA
  - 易于集成到8/16/32位系统中去
  - 基于Wi-Fi标准，802.11b解决方案，提供1 & 2Mb/s的数据传输速率
  - 完全支持Ad-Hoc和AP模式
  - 通过FCC和IC认证，并经过ETSI测试
  - 内置天线（MRF24WB0MA版）
  - 支持外部天线（MRF24WB0MB版）
  - 提供内建的WEP、WPA-PSK和WPA2-PSK安全模式
  - 支持Microchip TCPIP协议栈

产品适用性

- 适用的产品类型
  - 数据传输量少
  - 非连续传输
  - 数据不是高带宽的流数据

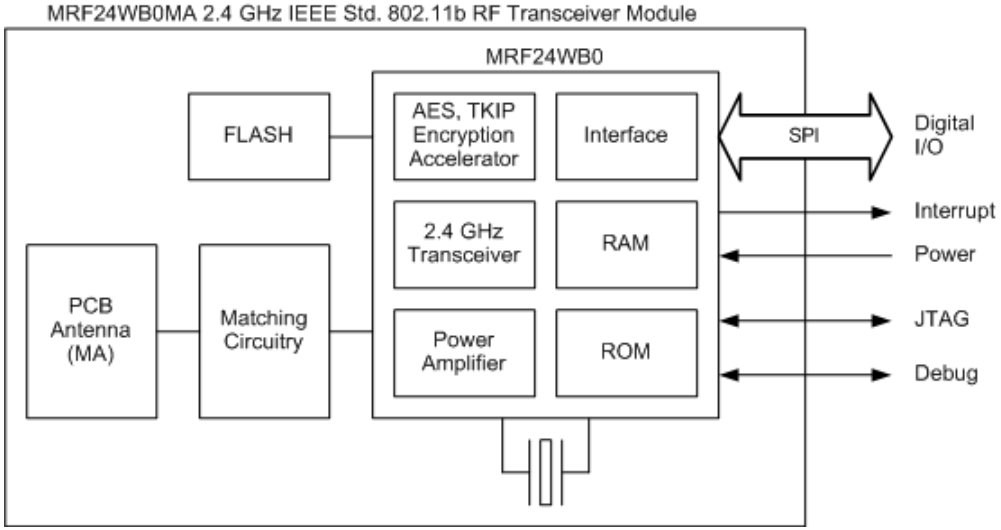






Microchip Wi-Fi®模块

MRF24WB0MA/B



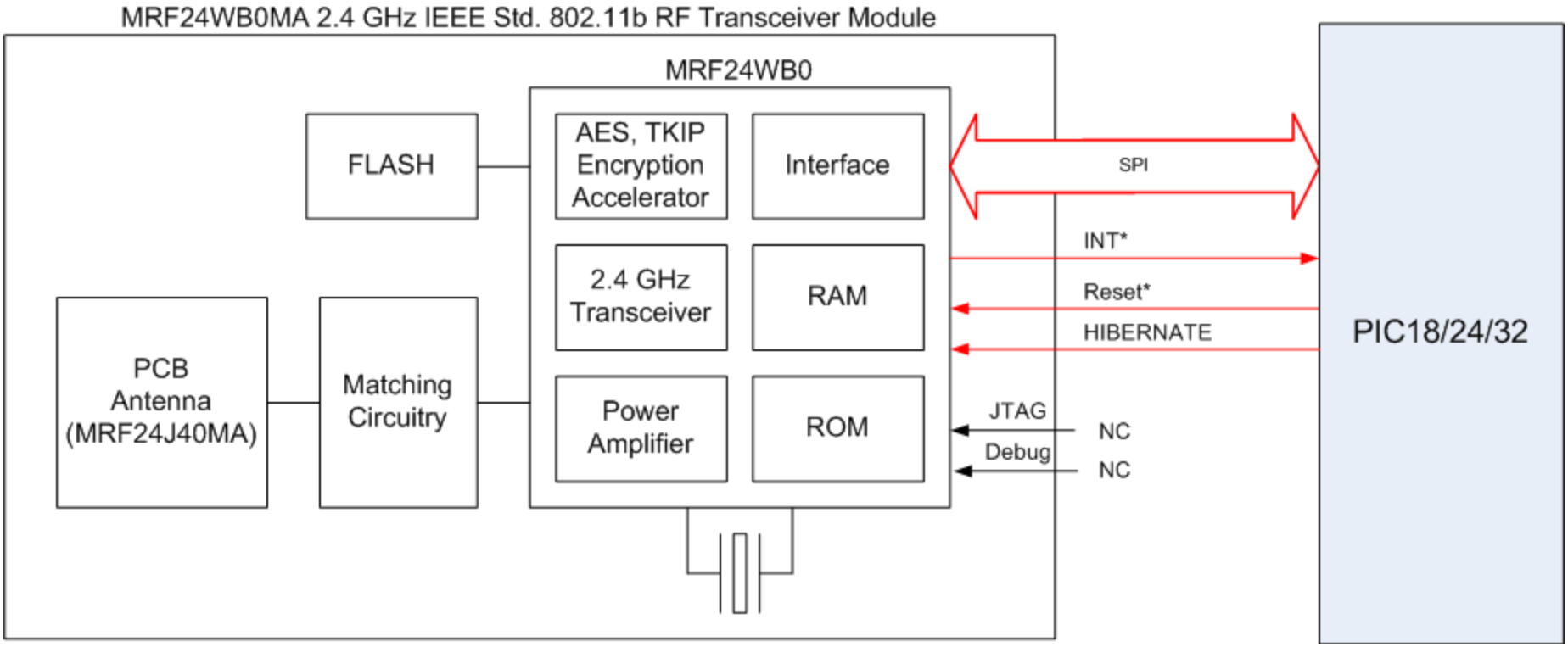
AC164136-4

用于开发的**PICtail™**

- PICtail连接器
- PICtail Plus连接器

Microchip Wi-Fi®模块

连接至PIC® MCU



产品版本

	第1代	第2代
模块	ZG2100MC/ZG2101MC	MRF24WB0MA/MB
产品规划	仅限量产	建议新开发的系统
协议栈	最高至V5.20	V5.25或更高
EZConfig	N/A	内置于应用程序（栈）
ZeroConfig	N/A	内置于应用程序（栈）
连接管理器	内置（栈）	内置（模块） 节省500B RAM
扫描	仅在空闲时	随时
栈的定制	函数修改	API接口
配置	TCPIPConfig.h	TCPIPConfig.h和WF_Config.h
PICtail™	AC164136-2	AC164136-4
文档	入门指南	栈的帮助文件

转换至MRF24WB

- 如果您处于量产阶段
  - 新部件将运行老的代码/栈
  - 不需要重做无意RF测试
- 处于开发阶段
  - 没有针对ZG产品的更新 —— 推荐变更为MRF24W
  - 使用v5.25，开发更为容易

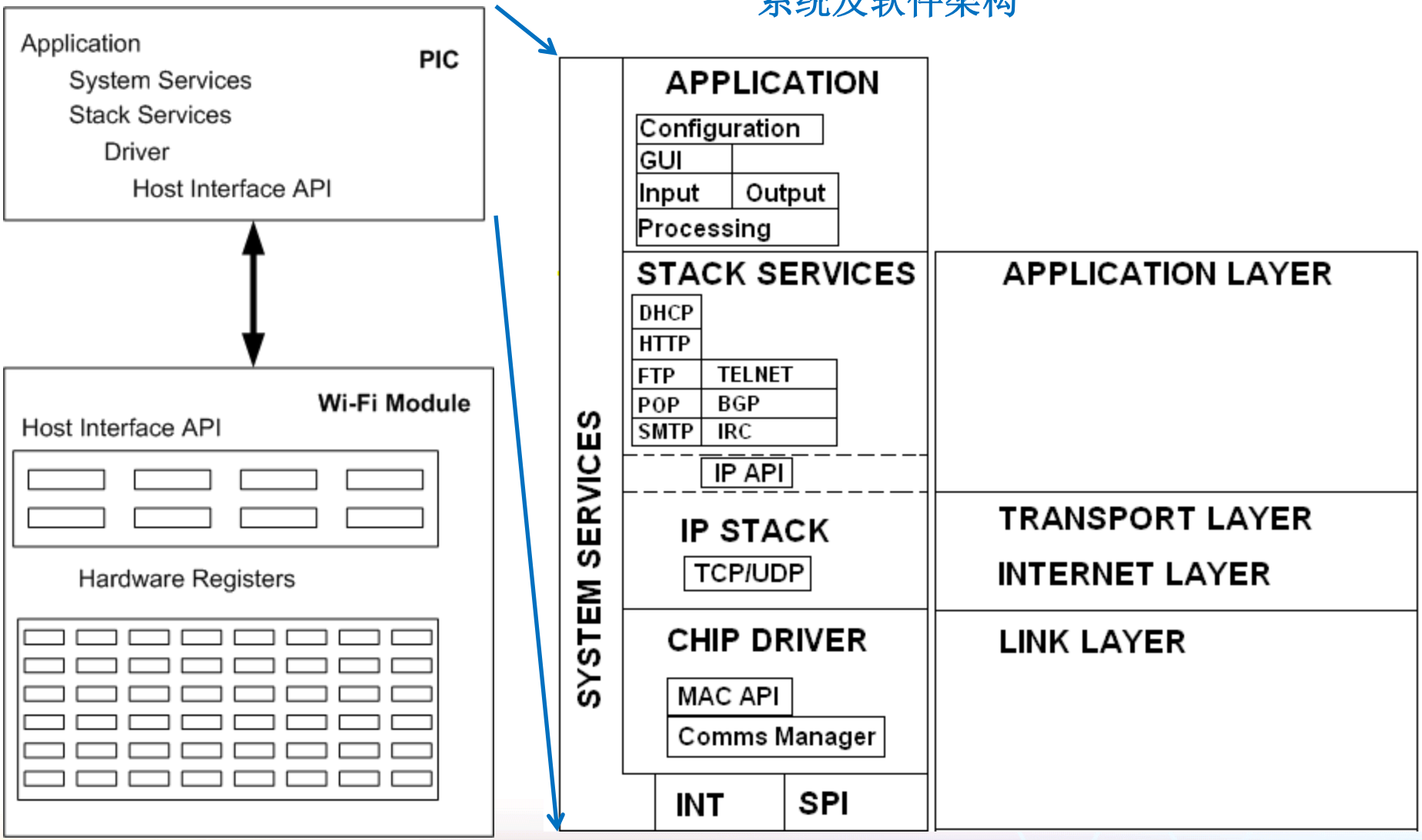
Microchip 802.11解决方案

802.11实现

产品应用	数据传输，浏览器，电子邮件客户机
应用服务	HTTP服务器，POP，FTP等
协议栈	TCP/IP，UDP/IP
驱动程序	ARP，MAC（扫描，关联，验证，连接状态）

Microchip 802.11解决方案

系统及软件架构



利用Microchip 802.11解决方案进行开发

向Microchip产品添加WiFi®功能

- 开始**Microchip TCPIP v5.25协议栈**
- 若干演示
  - TCPIP Wi-Fi Demo App
  - EZConfig Wi-Fi Demo App
  - Console Wi-Fi Demo App
- 提供可以直接开始使用的演示
- 适用于您应用的各种选项

DV240001

DM163024

AC164136-4

产品配置

- **TCPIP协议栈提供：**
  - 代码中的初始值定义
  - 配置应用的向导
  - “C”函数，可更改配置
  - 在运行时可动态更改配置的能力——提供范例应用

协议栈配置

TCPIPConfig.h

C:\MAL\May2010\TCPIP WiFi Demo App\TCPIPConfig.h

```
#define STACK_USE_UART              // Application demo using UART for IP address display and stack con
//#define STACK_USE_UART2TCP_BRIDGE // UART to TCP Bridge application example
//#define STACK_USE_IP_GLEANING
#define STACK_USE_ICMP_SERVER       // Ping query and response capability
//#define STACK_USE_ICMP_CLIENT     // Ping transmission capability
//#define STACK_USE_HTTP_SERVER     // Old HTTP server
#define STACK_USE_HTTP2_SERVER      // New HTTP server with POST, Cookies, Authentication, etc.
//#define STACK_USE_SSL_SERVER      // SSL server socket support (Requires SW300052)
//#define STACK_USE_SSL_CLIENT      // SSL client socket support (Requires SW300052)
#define STACK_USE_AUTO_IP           // Dynamic link-layer IP address automatic configuration protocol
#define STACK_USE_DHCP_CLIENT       // Dynamic Host Configuration Protocol client for obtaining IP addr
//#define STACK_USE_DHCP_SERVER     // Single host DHCP server
```

C:\MAL\May2010\TCPIP WiFi Demo App\TCPIPConfig.h

```
#define MY_DEFAULT_HOST_NAME        "MCHPBOARD"

#define MY_DEFAULT_MAC_BYTE1        (0x00)  // Use the default of
#define MY_DEFAULT_MAC_BYTE2        (0x04)  // 00-04-A3-00-00-00 if using
#define MY_DEFAULT_MAC_BYTE3        (0xA3)  // an ENCX24J600 or MRF24WB0M
#define MY_DEFAULT_MAC_BYTE4        (0x00)  // and wish to use the internal
#define MY_DEFAULT_MAC_BYTE5        (0x00)  // factory programmed MAC
#define MY_DEFAULT_MAC_BYTE6        (0x00)  // address instead.

#define MY_DEFAULT_IP_ADDR_BYTE1    (169ul)
#define MY_DEFAULT_IP_ADDR_BYTE2    (254ul)
#define MY_DEFAULT_IP_ADDR_BYTE3    (1ul)
#define MY_DEFAULT_IP_ADDR_BYTE4    (1ul)

#define MY_DEFAULT_MASK_BYTE1       (255ul)
#define MY_DEFAULT_MASK_BYTE2       (255ul)
#define MY_DEFAULT_MASK_BYTE3       (0ul)
#define MY_DEFAULT_MASK_BYTE4       (0ul)
```

更改MAC

注：00:04:A3:00:00:00

WiFi®配置

WF_Config.h

C:\MAL\May2010\TCPIP WiFi Demo App\WF_Config.h

```
/*-----------------------------------------------*/
/* Default settings for Connection Management */
/*-----------------------------------------------*/
#define MY_DEFAULT_SSID_NAME                "MicrochipDemoAP"

#define MY_DEFAULT_NETWORK_TYPE             WF_INFRASTRUCTURE   /* WF_INFRASTRUCTURE or WF_ADHOC     */

#define MY_DEFAULT_SCAN_TYPE                WF_ACTIVE_SCAN      /* WF_ACTIVE_SCAN or WF_PASSIVE_SCAN */

#define MY_DEFAULT_CHANNEL_LIST             {1,6,11}            /* use {} to scan all channels       */

#define MY_DEFAULT_LIST_RETRY_COUNT         (3)

#define MY_DEFAULT_EVENT_NOTIFICATION_LIST  (WF_NOTIFY_CONNECTION_ATTEMPT_SUCCESSFUL  |          \
                                             WF_NOTIFY_CONNECTION_ATTEMPT_FAILED      |          \
                                             WF_NOTIFY_CONNECTION_TEMPORARILY_LOST    |          \
                                             WF_NOTIFY_CONNECTION_PERMANENTLY_LOST    |          \
                                             WF_NOTIFY_CONNECTION_REESTABLISHED)

#define MY_DEFAULT_PS_POLL                  WF_DISABLED         /* WF_DISABLED or WF_ENABLED */

#define MY_DEFAULT_WIFI_SECURITY_MODE       WF_SECURITY_OPEN

//#define USE_MRF24W_HOST_BUFFER

//#define STACK_USE_EZ_CONFIG
//#define EZ_CONFIG_SCAN
//#define EZ_CONFIG_STALL
//#define EZ_CONFIG_STORE
```

WiFi®配置

WF_Config.h

C:\MAL\May2010\TCPIP WiFi Demo App\WF_Config.h

```
/*****************************************************************************/
/*****************************************************************************/
/*                              WIFI SECURITY COMPILE-TIME DEFAULTS           */
/*****************************************************************************/
/*****************************************************************************/
// Security modes available on WiFi network:
//   WF_SECURITY_OPEN                       : No security
//   WF_SECURITY_WEP_40                     : WEP Encryption using 40 bit keys
//   WF_SECURITY_WEP_104                    : WEP Encryption using 104 bit keys
//   WF_SECURITY_WPA_WITH_KEY               : WPA-PSK Personal where binary key is given to MRF24WB0M
//   WF_SECURITY_WPA_WITH_PASS_PHRASE       : WPA-PSK Personal where passphrase is given to MRF24WB0M and it c
//   WF_SECURITY_WPA2_WITH_KEY              : WPA2-PSK Personal where binary key is given to MRF24WB0M
//   WF_SECURITY_WPA2_WITH_PASS_PHRASE      : WPA2-PSK Personal where passphrase is given to MRF24WB0M and it
//   WF_SECURITY_WPA_AUTO_WITH_KEY          : WPA-PSK Personal or WPA2-PSK Personal where binary key is given
//                                            connect at highest level AP supports (WPA or WPA2)
//   WF_SECURITY_WPA_AUTO_WITH_PASS_PHRASE : WPA-PSK Personal or WPA2-PSK Personal where passphrase is given
//                                            calculates the binary key and connects at highest level AP sup
```

安全性类型

WiFi®配置

WF_Config.h

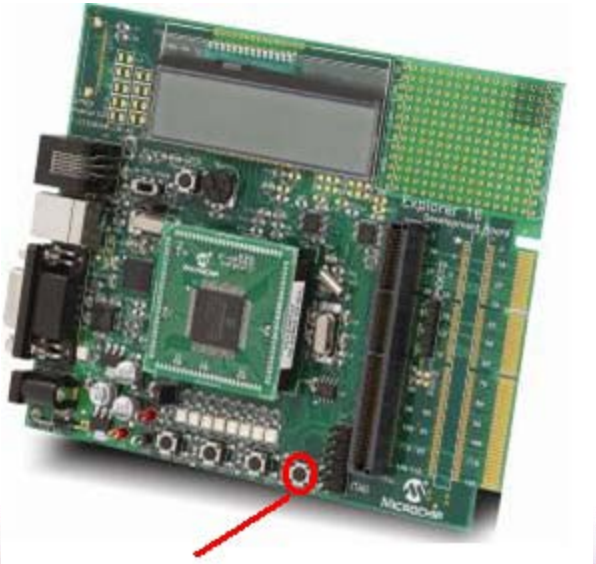
C:\MAL\May2010\TCPIP WiFi Demo App\WF_Config.h

```
// Default pass phrase used for WF_SECURITY_WPA_WITH_PASS_PHRASE and
// WF_SECURITY_WPA2_WITH_PASS_PHRASE security modes
#define MY_DEFAULT_PSK_PHRASE                "Microchip 802.11 Secret PSK Password"


// If using security mode of WF_SECURITY_WPA_WITH_KEY or WF_SECURITY_WPA2_WITH_KEY, then this section
// must be set to  match the key for MY_DEFAULT_SSID_NAME and MY_DEFAULT_PSK_PHRASE
// combination.  The values below are derived from the SSID "MicrochipDemoAP" and the pass phrase
// "Microchip 802.11 Secret PSK Password".
// The tool at http://www.wireshark.org/tools/wpa-psk.html can be used to generate this field.
#define MY_DEFAULT_PSK "\
\x86\xC5\x1D\x71\xD9\x1A\xAA\x49\
\x40\xC8\x88\xC6\xE9\x7A\x4A\xD5\
\xE5\x6D\xDA\x44\x8E\xFB\x9C\x0A\
\xE1\x47\x81\x52\x31\x1C\x13\x7C"


//------------------------------------------------------------------------------
// Default WEP keys used in WF_SECURITY_WEP_40  and WF_SECURITY_WEP_104 security mode
//------------------------------------------------------------------------------
#define MY_DEFAULT_WEP_PHRASE            "WEP Phrase"

// string 4 40-bit WEP keys -- corresponding to passphraseof "WEP Phrase"
#define MY_DEFAULT_WEP_KEYS_40 "\
\x5a\xfb\x6c\x8e\x77\
\xc1\x04\x49\xfd\x4e\
\x43\x18\x2b\x33\x88\
\xb0\x73\x69\xf4\x78"
```

安全性细节

配置Infrastructure

- 演示应用已经预先配置好
- 要更改初始值，请擦除EEPROM
- 可更改：安全性和网络



TCPIP Wi-Fi®演示应用

AdHoc演示

EZConfig和Bonjour演示

配置选项

- 支持DHCP、客户机及服务器
- AdHoc
- 安全性（WEP，WPA，WPA2）
- 功率模式
- 冬眠讨论

DHCP

- 主机控制协议 = IP寻址
  - 也称为“获取IP地址”
  - 动态（DHCP）或静态（Static）
  - 动态寻址需要服务器
    - **演示开始时使用缺省的“静态”IP地址169.254.1.1（为什么是这个地址？）**
    - **如果启用了DHCP客户机，并且得到响应的话，地址将会改变**
    - **没有启用DHCP客户机，就意味着静态IP**
- 静态时注意节能事项

安全性

- 隐藏SSID并不具备安全特性
- 仅使用WEP索引0
- WEP-40（64位WEP）需要5个ASCII字符，等同于10个十六进制值
- WEP-104（128位WEP）需要13个ASCII字符，等同于26个十六进制值
- 开放的WEP ——它更安全，不进行“验证”——您不知道输入的密钥是否错误，只知道双方无法互相理解
- WPA需要一个用SSID计算得出的密钥
- 使用推荐的站点进行密钥计算（WEP/WPA）

AdHoc网络

- 开始时在EZConfig中使用
- 非常适合临时网络
- AdHoc模式只提供WEP加密方法
- iPhone是从最初就支持它的惟一的智能手机
- 确保MRF24WB始终启动AdHoc网络，或者把所有客户机锁定为2 Mbps

功率模式

- **3种模式**
  - 主动发送，主动接收
    - **185mA，85mA**
  - 休眠（PS，节能模式）
    - **250uA**
  - 冬眠
    - **0.1uA**
    - 支持

MASTERs CONFERENCE 2010

创建客户机和服务器应用

应用类型

基于服务器的

- 在网络上绝大多数时间是被动的
- 产品实现需求
- 当前演示
- 通用
- 到现场最快
- 由于仅对原始数据进行服务，降低了带宽和存储空间需求

基于客户机的

- 主动性平台
- 运行以便接入网络
- 利用接收到的数据，用产品“做点什么”

- 可以是混合的 —— 即，从服务器读取表单，填充数据，然后把数据上传回服务器

应用类型

基于服务器的

- 在网络上绝大多数时间是被动的
- 产品实现需求
- 当前演示
- 通用
- 到现场最快

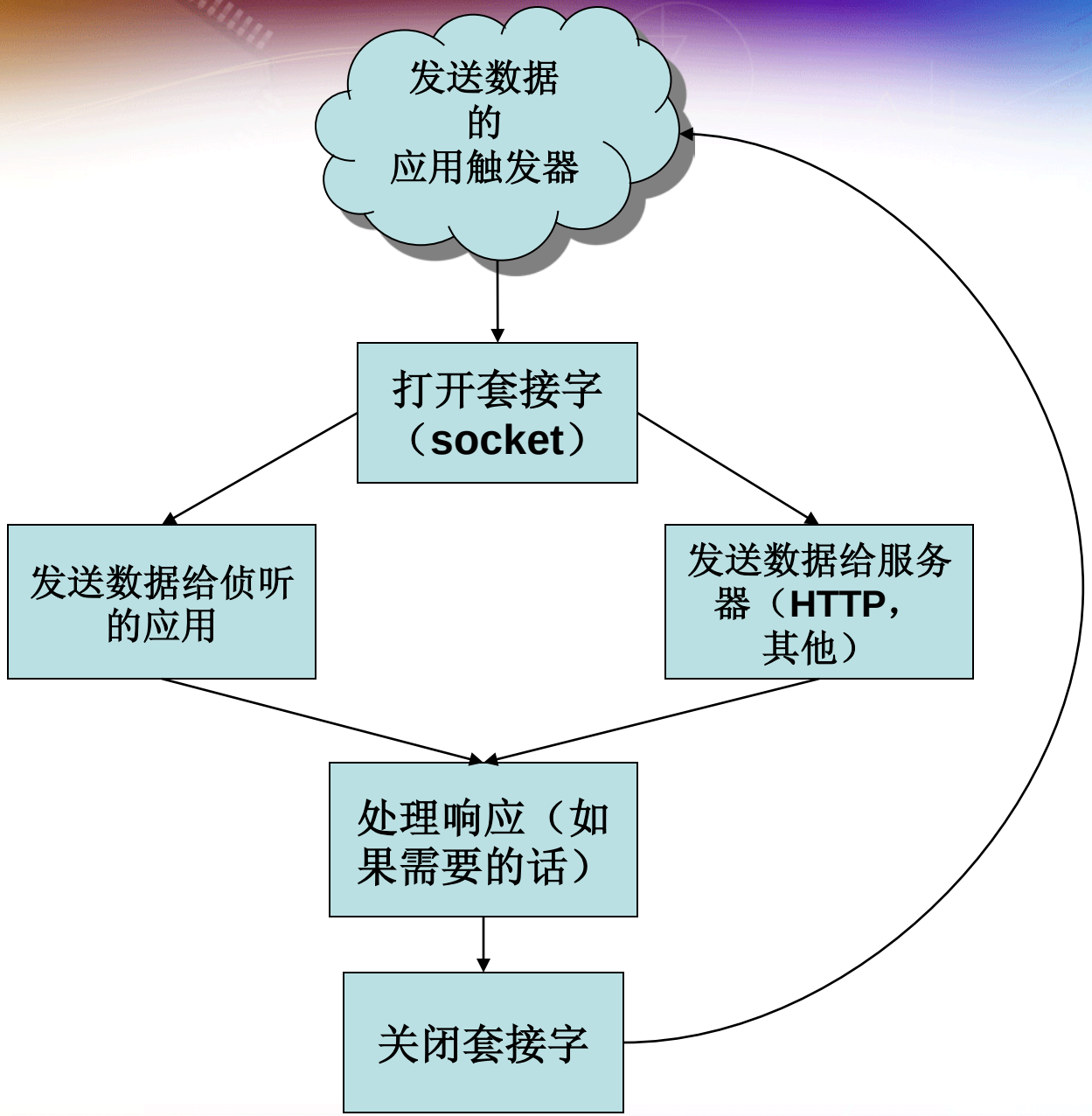
基于客户机的

- 主动性平台
- 运行以便接入网络
- 利用接收到的数据，用产品“做点什么”

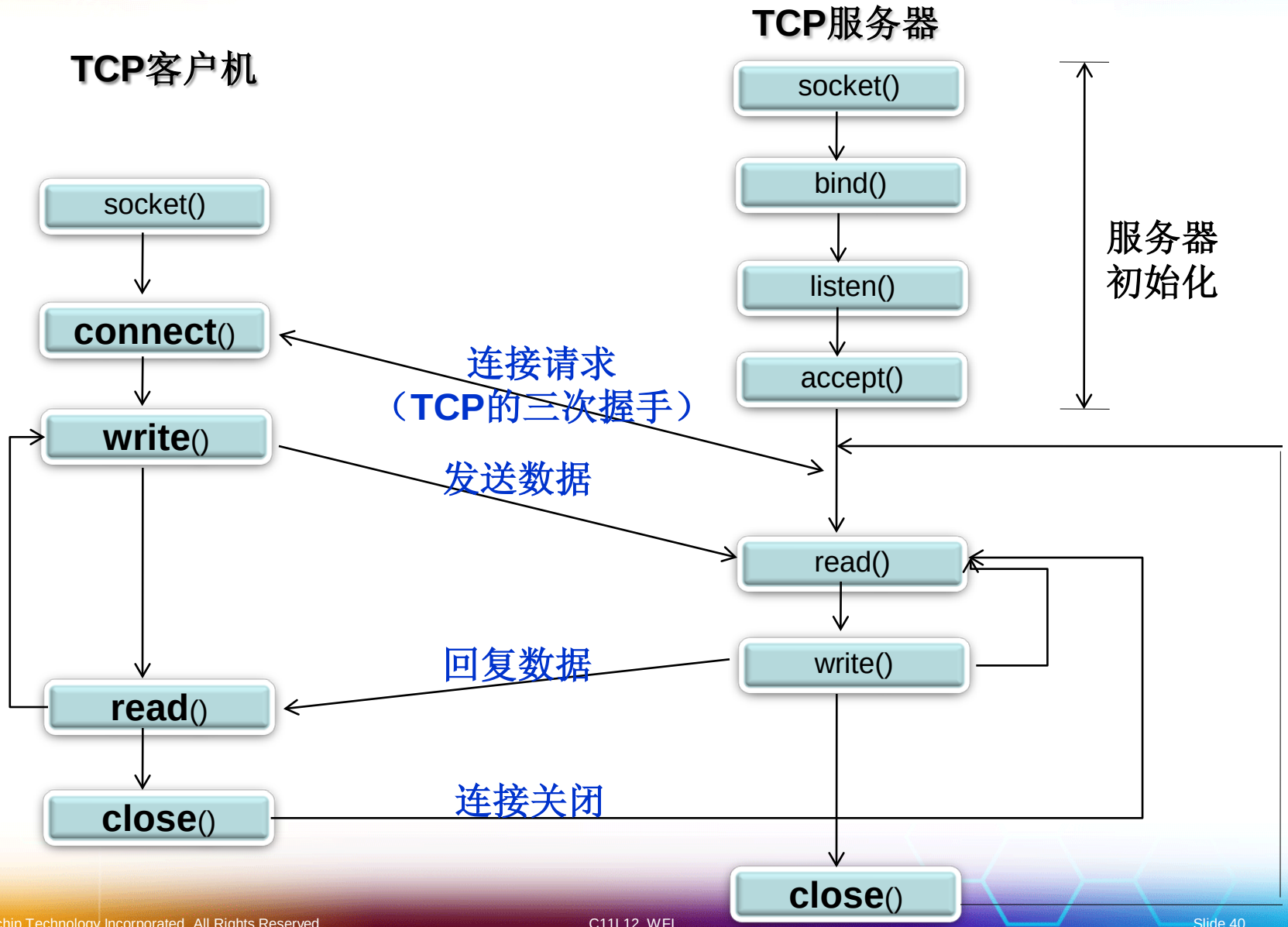
- 由于仅对原始数据进行服务，降低了带宽和存储空间需求
- 可以是混合的，从服务器读取表单，判断并把数据上传回服务器

客户机的编程

- 需要下列其中之一：
  - 定制应用，对具体端口进行侦听
  - 远程服务器，用来处理TCP消息（如HTTP）
- 如果要求的话，处理返回的数据
- 使用GenericTCPClient演示作为着手点



客户机—服务器架构



启用GenericTCPClient

C:\MAL\v2010-04-28\TCPIP WiFi Demo App\TCPIPConfig.h*

```
76  //#define STACK_USE_SSL_CLIENT          // SSL client socket support (Re
77  #define STACK_USE_AUTO_IP               // Dynamic link-layer IP address
78  #define STACK_USE_DHCP_CLIENT           // Dynamic Host Configuration Pr
79  //#define STACK_USE_DHCP_SERVER         // Single host DHCP server
80  //#define STACK_USE_FTP_SERVER          // File Transfer Protocol (old)
81  //#define STACK_USE_SMTP_CLIENT         // Simple Mail Transfer Protocol
82  //#define STACK_USE_SNMP_SERVER         // Simple Network Management Pro
83  //#define STACK_USE_TFTP_CLIENT         // Trivial File Transfer Protoco
84  #define STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE     // HTTP Client example i
85  //#define STACK_USE_GENERIC_TCP_SERVER_EXAMPLE // ToUpper server exampl
86  //#define STACK_USE_TELNET_SERVER           // Telnet server
87  #define STACK_USE_ANNOUNCE              // Microchip Embedded Ethernet D
88  #define STACK_USE_DNS                   // Domain Na
89  //#define STACK_USE_DNS_SERVER          // Domain Na
90  #define STACK_USE_NBNS                  // NetBIOS N
91  #define STACK_USE_REBOOT_SERVER         // Module for resetting this PIC
92  //#define STACK_USE_SNTP_CLIENT         // Simple Network Time Protocol
93  //#define STACK_USE_UDP_PERFORMANCE_TEST    // Module for testing UDP TX
94  #define STACK_USE_TCP_PERFORMANCE_TEST  // Module for testing TCP TX per
95  //#define STACK_USE_DYNAMICDNS_CLIENT       // Dynamic DNS client update
96  //#define STACK_USE_BERKELEY_API            // Berekely Sockets APIs are
97  //#define STACK_USE_ZEROCONF_LINK_LOCAL // Zeroconf IPv4 Link-Local Addr
98  //#define STACK_USE_ZEROCONF_MDNS_SD        // Zeroconf mDNS and mDNS se
99
100
```

Enable this define

启用GenericTCPClient

C:\MAL\v2010-04-28\TCPIP WiFi Demo App\TCPIPConfig.h*

```
274         // FIFO can be zero if desired.
275         #define TCP_CONFIGURATION
276         ROM struct
277         {
278             BYTE vSocketPurpose;
279             BYTE vMemoryMedium;
280             WORD wTXBufferSize;
281             WORD wRXBufferSize;
282         } TCPSocketInitializer[] =
283         {
284             {TCP_PURPOSE_GENERIC_TCP_CLIENT, TCP_ETH_RAM, 512, 512},
285             //{TCP_PURPOSE_GENERIC_TCP_SERVER, TCP_ETH_RAM, 20, 20},
286             //{TCP_PURPOSE_TELNET, TCP_ETH_RAM, 200, 150},
287             //{TCP_PURPOSE_TELNET, TCP_ETH_RAM, 200, 150},
288             //{TCP_PURPOSE_TELNET, TCP_ETH_RAM, 200, 150},
289             //{TCP_PURPOSE_FTP_COMMAND, TCP_ETH_RAM, 100, 40},
290             //{TCP_PURPOSE_FTP_DATA, TCP_ETH_RAM, 0, 128},
291             {TCP_PURPOSE_TCP_PERFORMANCE_TX, TCP_ETH_RAM, 200, 1},
292             //{TCP_PURPOSE_TCP_PERFORMANCE_RX, TCP_ETH_RAM, 40, 1500},
293             //{TCP_PURPOSE_UART_2_TCP_BRIDGE, TCP_ETH_RAM, 256, 256},
294             {TCP_PURPOSE_HTTP_SERVER, TCP_ETH_RAM, 1000, 1000},
295             {TCP_PURPOSE_HTTP_SERVER, TCP_ETH_RAM, 1000, 1000},
296             //{TCP_PURPOSE_DEFAULT, TCP_ETH_RAM, 1000, 1000},
297             //{TCP_PURPOSE_BERKELEY_SERVER, TCP_ETH_RAM, 25, 20},
298             //{TCP_PURPOSE_BERKELEY_SERVER, TCP_ETH_RAM, 25, 20},
```

Enable this entry and assign a meaningful Tx and Rx buffer size

需要把TX和RX缓冲区设置于合理的值（但要在TCP_ETH_RAM_SIZE给出的限制范围内）

GenericTCPClient.c

C:\MAL\v2010-04-28\TCPIP WiFi Demo App\GenericTCPClient.c

```
100     ********************************************************************/
101 void GenericTCPClient(void)
102 {
103     BYTE                i;
104     WORD                w;
105     BYTE                vBuffer[9];
106     static DWORD        Timer;
107     static TCP_SOCKET   MySocket = INVALID_SOCKET;
108     static enum _GenericTCPExampleState
109     {
110         SM_HOME = 0,
111         SM_SOCKET_OBTAINED,
112         SM_PROCESS_RESPONSE,
113         SM_DISCONNECT,
114         SM_DONE
115     } GenericTCPExampleState = SM_DONE;
116
117     switch(GenericTCPExampleState)
118     {
119         case SM_HOME:
120             // Connect a socket to the remote TCP server
121             MySocket = TCPOpen((DWORD)&ServerName[0], TCP_OPEN_RAM_HOST, ServerPort, TCP_PUI
122
123             // Abort operation if no TCP socket of type TCP_PURPOSE_GENERIC_TCP_CLIENT is av
124             // If this ever happens, you need to go add one to TCPIPConfig.h
125             if(MySocket == INVALID_SOCKET)
126                 break;
127
128             #if defined(STACK_USE_UART)
129             putrsUART((ROM char*)"\r\n\r\nConnecting using Microchip TCP API...\r\n");
130             #endif
131
132             GenericTCPExampleState++;
133             Timer = TickGet();
134             break;
135
136         case SM_SOCKET_OBTAINED:
137             // Wait for the remote server to accept our connection request
138             if(!TCPIsConnected(MySocket))
139             {
140                 // Time out if too much time is spent in this state
141                 if(TickGet()-Timer > 5*TICK_SECOND)
142                 {
143                     // Close the socket so it can be used by other modules
```

应用触发器

- 主应用需要触发GenericTCPClient状态机，以打开连接
- 对于演示而言，它在等待按钮按下

打开套接字

- 使用**TCPOpen()**打开套接字，而后使用**TCPIsConnected()**等待套接字连接上

发送数据

- **TCPIsPutReady()将返回可用于发送的缓冲空间**
  - 需要把协作式多任务状态机前进至下一状态，以便在需要时释放更多的空间
- **对TCPPut...()的任何调用都将向套接字发送数据**
- **如果在另一端使用定制的侦听程序，要发送数据的格式需要与期望的相一致**

发送数据HTTP

- 对于HTTP，数据需要针对恰当的HTTP头进行格式化
- 内容类型也需要与您要发送给远程节点的相匹配

处理呼入数据

- 如果希望数据被返回，**TCPIsGetReady()**将返回等待接收的字节数
- 使用**TCPGet...()**函数获取返回数据
- 最终用户需要编写函数来解析返回数据并获取所需信息

关闭连接

- 一旦完成，调用**TCPDisconnect()**来关闭连接
- 然后，**GenericTCPClient**状态机可以返回，等待来自更高层应用的激励







下一步

让您的产品连上互联网

工程步骤

- 使用TCPIP WiFi® EZConfig演示应用开始
- LED/按钮示例是您的I/O
- 更新Web页面，添加更多的用户控制页面

下一步

- 创建PC、iPhone、Droid客户机应用，而不是对所有数据进行服务
  - 这将显著降低带宽需求
- 添加客户机服务，通过无线方式获取并解析天气信息

今天我们讨论了：

- 什么是嵌入式系统的Wi-Fi®
- 如何使用Microchip 802.11解决方案
- 如何运行Microchip演示
- 如何配置无线网络
- 利用Microchip演示，把您的产品接入无线网络

更多资源

Microchip硬件/软件

- http://www.microchip.com/wifi

数据手册/应用笔记

- http://www.microchip.com/wifi

支持

- http://support.microchip.com

协议栈、示例和入门文档

- START>Microchip>TCPIP Stack v5.25>TCPIP Stack Help

书籍

- 802.11 Wireless Networks, The Definitive Guide. M.Gast. O'Reilly 2005.
- The IEEE 802.11 Handbook: A Designer's Companion. Ohara; Petrick. IEEE 2005.

商标

Microchip的名称和徽标组合、Microchip徽标、dsPIC、KeeLoq、KeeLoq徽标、MPLAB、PIC、PICmicro、PICSTART、PIC32徽标、rfPIC及UNI/O均为Microchip Technology Incorporated在美国和其他国家或地区的注册商标。

FilterLab、Hampshire、HI-TECH C、Linear Active Thermistor、MXDEV、MXLAB、SEEVAL和The Embedded Control Solutions Company均为Microchip Technology Incorporated在美国的注册商标。

Analog-for-the-Digital Age、Application Maestro、CodeGuard、dsPICDEM、dsPICDEM.net、dsPICworks、dsSPEAK、ECAN、ECONOMONITOR、FanSense、HI-TIDE、In-Circuit Serial Programming、ICSP、Mindi、MiWi、MPASM、MPLAB Certified徽标、MPLIB、MPLINK、mTouch、Omniscient Code Generation、PICC、PICC-18、PICDEM、PICDEM.net、PICkit、PICtail、REAL ICE、rfLAB、Select Mode、Total Endurance、TSHARC、UniWinDriver、WiperLock和ZENA均为Microchip Technology Incorporated在美国和其他国家或地区的商标。

SQTP是Microchip Technology Incorporated在美国的服务标记。

在此提及的所有其他商标均为各持有公司所有。